Efficient Traversability Mapping for Service Robots Using a Point-cloud Fast Filter

Carlos Medina Sánchez[1], Matteo Zella[1], Jesús Capitan[2], Pedro J. Marron[1]

***Abstract*— This paper proposes *Point-cloud Fast Filter* (PFF), an algorithm to process efficiently 3D data from point-cloud sensors in order to build traversability maps for service robots. Our method is intended to be integrated with a 2D mapping algorithm, enhancing 2D standard maps with enough traversability information for robot navigation in indoor structured environments. The method is agnostic to the 3D sensor or mapping algorithm used, and keeps computational requirements low. Thus, we enable middle-class computers and a wide variety of sensors to be employed for service robots, reducing the costs of the platform. We evaluate the performance of PFF with different 3D sensors on a real robot and its impact on mapping, comparing it with alternative 2D and 3D mapping approaches.**

I. Introduction

Service robots have become a reality in the last decade. The enhancement in the robustness of robotics platforms has enabled robots to operate in a long-term (more than one day) fashion for elderly care [1], office delivery tasks [2] or assistance in hospitals [3]. In these indoor environments, safe and precise robot localization and navigation are of uppermost importance. Moreover, in long-term operation, the addition of 3D information to determine traversability maps for the robots is crucial. In fact, in the case of permanent non-traversable objects such as stairs, as well as movable ones like tables or chairs, 2D sensors might fail to detect their presence, thus affecting the ability of robots to operate in such scenarios. However, 3D mapping and localization algorithms (Sec. II) are known to be computationally intensive tasks in general.

We are interested in investigating efficient mapping approaches that suffice for robot navigation in office-like scenarios (multi room environments with office furniture: desks, chairs, etc.). This implies integrating information from 3D sensors, while constraining the processing significantly, thus making the approach practical for service robots of limited resources and cost, such as the common TurtleBot 2 platforms shown in Fig. 1. To achieve this, we introduce the *Point-cloud Fast Filter* (PFF) to process efficiently 3D data from point-cloud sensors in order to build robot traversability maps, or so-called 2.5D maps (Sec. III). First, 3D data are filtered out depending on robot physical constraints. Then, a method to recognize stairs or similar non-traversable areas is applied. In doing this, we design a simple, yet effective algorithm aimed at running on middle-class (see specs of the BM in Sec. IV) computing machines.

[1]University of Duisburg-Essen. Essen, 45127, Germany. carlos.medina-sanchez@uni-due.de, matteo.zella@uni-due.de, pjmarron@uni-due.de
[2]University of Seville. Seville, 41092, Spain. jcapitan@us.es

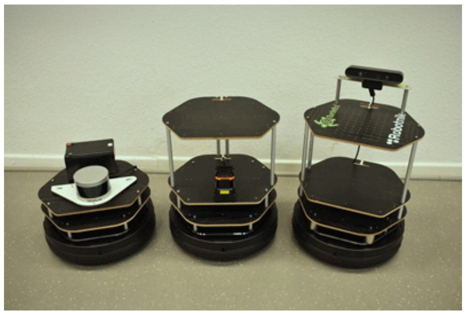


Fig. 1: Example of our Turtlebot 2 platforms used as service robots in office-like scenarios.

This paper, therefore, offers the following contributions:

- The design of a practical algorithm (PFF) able to efficiently extract relevant data for mapping and localization from point-cloud sensors, which can be executed on middle-class machines;
- A method agnostic to the sensor or mapping algorithm used, as PFF can be integrated with different mapping algorithms and sensors (e.g., LiDARs or RGB-D cameras) as long as their interface provides point-cloud data;
- The analysis of the impact of different 3D sensors on the behavior of PFF, highlighting the trade-offs between alternative system configurations.

We evaluate our approach (Sec. IV) using well-known 3D sensors to assess its efficiency for traditional platforms. We also demonstrate the overall performance by running a mapping algorithm on a real robot operating in an indoor environment. Based on the experience gathered in this work, we identify possible directions for future work (Sec. V).

II. Related Work

Indoor robot mapping and localization is a well-known and active research topic, due to the variety of aspects involved. In particular, the interplay between software and hardware with respect to the application requirements defines a complex design space. In this section, we highlight the relevant aspects of interest for our work.

A. Sensors for Mapping

Before discussing the algorithmic aspects involved in robot mapping, we identify the different classes of sensors that can be exploited to perceive a robot's surroundings. For indoor scenarios, three main sensor categories can be identified: 2D LiDARs, depth cameras, and 3D LiDARs. An overview of the characteristics of typical sensors from each category is reported in Table I. We highlight the differences between the various sensors to later analyze the impact of each sensing technology on the resulting behavior of our approach. In any case, our main goal is to design an efficient solution that is agnostic to the specific sensor employed.

A first interesting characteristic distinguishing LiDARs from cameras is the practical inability to perform facial recognition, thus better protecting the privacy of people walking in the environment. According to their sampling time, and assuming a social environment with people moving at an average speed of $1\ m/s$, the 2D LiDAR would be able to capture displacements of $5\ cm$ per sample, the 3D LiDAR $10\ cm$ per sample, and the camera $3.4\ cm$. Therefore, the difference in sampling rate is not significant. With respect to the Field of View (FoV), LiDARs can reach a horizontal FoV of up to 360°. However, a 2D sensor is clearly limited to only a plane, while the VLP-16 (3D LiDAR) has a vertical field of view of 30°, which is already sufficient for many applications in service robotics; furthermore, even if significantly more expensive, it can reach up to $100m$. In comparison, depth cameras are cheaper while still offering 3D perception data; though, their FoV is significantly limited.

B. Traversability Mapping

For *Unmanned Ground Robots*, employed as indoor service robots, it is common to work with 2D information from laser-based sensors [4]. To handle this type of information there is a wide spectrum of algorithms able to build maps for 2D robot navigation [5]. However, depending on the type of environment, 2D information may not suffice to create reliable maps that guarantee safe robot navigation. Therefore, enhanced methods using richer information have been proposed. One option is to integrate measurements from multiple 2D lasers located at different robot's positions and orientations [6]. Other works have proposed hardware improvements adding degrees of freedom to 2D sensors by using servo motors, which allow the robot to capture 3D information [7].

A simpler solution from the hardware point of view is to use sensors that directly provide 3D data, like depth cameras (RGB-D) and 3D LiDARs. These had an important growth thanks to the appearance of *Unmanned Aerial Vehicles* [8]. For indoor applications, depth cameras are more used since they are cheaper and their shorter ranges are typically sufficient. Some approaches use only depth information to build 3D maps, while others combine it with color images for more realistic maps [9], [10]. In [11], the *Fast Sampling Plane Filtering* algorithm is presented for indoor localization with a depth camera; something similar was proposed by [12] but using a 2D laser with a servo motor to produce 3D information.

One of the main limitations of depth cameras is their reduced range and field of view. This is why 3D LiDARs are preferred for outdoor applications, as they can easily reach more than 50 meters perception range. Thus, methods for 3D mapping with LiDARs like *LOAM* [13] are becoming

TABLE I: Comparison of different sensors for mapping.

Sensor	Privacy	Sample time	FoV (H - V)	Range	Price (USD)
2D LiDAR					
RPLIDAR A3 [19]	+ +	50 ms	360°-N/A	25 m	500
Depth camera (RGB-D)					
Astra Camera [20]	- -	33 ms	60°-49.5°	8 m	150
3D LiDAR					
VLP-16 [21]	+ +	100 ms	360°-30°	100 m	4000

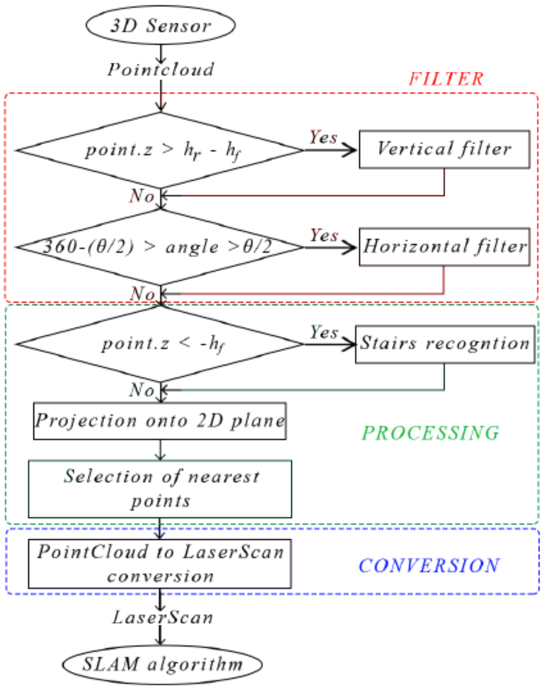


Fig. 2: PFF algorithm divided into its three stages: filtering, processing and conversion. See Fig. 3 for h_f, h_r and θ.

common in aerial robotics as well as in autonomous car driving [14], [15]. Even though these methods can provide highly accurate maps, they are computationally demanding. Our method lies in the category of the so-called 2.5D mapping approaches, where 2D maps are built with traversability information focused not on the type of sensor [16], but rather on the type of information (point-cloud). These methods integrate 3D information into usual 2D maps in order to determine the areas that cannot be traversed by the robot. This may still imply heavy processing depending on the algorithm [17] or if multiple sensors are used for this task [18].

In this work, we focus on identifying an efficient solution with limited computation requirements that is, nonetheless, enough for indoor navigation in office-like scenarios. By achieving this, we enable middle-class computers to be employed for service robots, allowing a variety of sensors for perception, and thus controlling the costs of the employed platforms. At the same time, our approach frees computation resources that can be exploited to run further algorithms and services.

III. Point-cloud Fast Filter

In this section, we introduce our solution to process 3D data in order to build traversability maps. Our approach is based on the definition of an efficient and practical filter, i.e., the *Point-cloud Fast Filter* (PFF). The structure of the PFF, shown in Fig. 2, is composed of several steps that can be grouped into three consecutive stages. The PFF receives 3D data in point-cloud format and the first stage involves a horizontal and vertical filtering of points that are outside the robot's range of interest. The result is another 3D point-cloud only with the points that match relevant obstacles for the robot movements. The second stage performs the processing of useful points to recognize stairs and to project the 3D information onto a 2D plane, keeping only the nearest points. The outcome is a 2D point-cloud data structure with one point per angle, which is then converted into laser scans in the last stage. This data type can be provided as input to traditional 2D SLAM algorithms. In the reminder of this section, we describe each stage individually.

A. Horizontal and Vertical Filter

First, PFF implements a vertical filter that divides the 3D points into three sections depending on their height. The first section contains all points placed at a height higher than the robot height, h_r. This information is filtered because it does not interfere with the robot navigation (Fig. 3(a), red section). The other sections cover all the remaining points, whose height value can affect the ability of the robot to traverse an area or not (Fig. 3(a), green and blue sections). These points will be further processed in the following steps.

After identifying the points vertically relevant for a traversability map, a horizontal filter is applied to preserve only information within a FoV of interest, given the robot task at hand and the sensor employed. The viewing angle is centered at $0°$ and opens symmetrically from $-\theta/2$ to $\theta/2$ (Fig. 3(b)); all points outside of this range are discarded. Note that θ cannot be greater than the actual sensor FoV, for example, for a Velodyne VLP-16 $360°$, for the Hokuyo UTM-30LN $270°$, and for the Astra Camera $60°$. This ability to dynamically adapt the FoV used is crucial to further decrease the number of points and focus the subsequent processing exclusively on the points of relevance for a specific task, e.g., to drive the robot in a map already known.

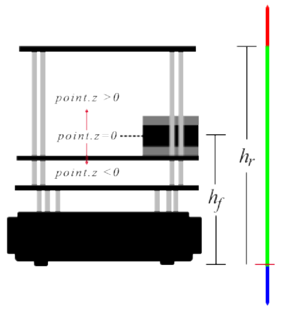
B. Processing

Once the points with impact on traversability are identified, particular care must be taken to recognize structural obstacles like stairs. For convenience, we focus in the following on the example of stairs, even though the approach is generally applicable to any areas with "holes" (no ramps) that the robot cannot visit. These cases involve the points whose height is less than the surface above which the robot moves, $-h_f$ (Fig. 3(a), blue section). Due to the vertical viewing angle of the sensor, a complete perception of the stair may not be possible. For this reason, an estimation needs to be made by interpolating the different points whose height is below the robot. If both the floor and the stair are within field of view, it is possible to easily understand where the stair starts and therefore move the points with a height lower than the floor (e.g., the red point in Fig. 4) to the front of the robot as to build a "virtual wall" (e.g., the blue point nearest to the robot in Fig. 4). If the sensor cannot perceive the floor, the points are replaced to build such a "wall" directly in front of the robot, as a precise positioning of the stair is not possible. It is worth noting that if the stair is ascending instead of descending, it will be treated as any other obstacle placed above ground, so no extra processing is required.

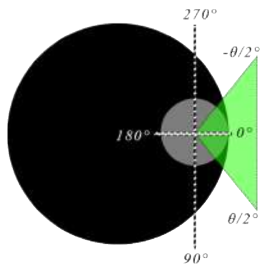
Now that all the points relevant for the robot dimensions and FoV are selected, they can be projected onto a 2D plane. Thus, the data can be used as input for traditional SLAM algorithms based on 2D sensor information. The last step in the processing stage selects, for each horizontal angle direction (with a resolution of R degrees), only the nearest point while discarding the others which are farther. For instance, if a 3D obstacle is within the robot FoV (see Fig. 5), only the nearest points are used, which are the first points that the robot would encounter if it had to move towards that obstacle (green points in Fig. 5). The result is a point-cloud on a 2D plane with one point per angle, using R as angular resolution. The nearest point in terms of horizontal distance from those at different heights is kept at each angle interval.

C. Point-cloud to Laser Scan Conversion

Once a 2D traversability map has been built, it is necessary to convert such data in the appropriate format to be handled by the SLAM algorithm of choice. For that, it is possible to



(a)



(b)

Fig. 3: Filtering of 3D data based on robot height and relevant FoV: (a) Vertical filter: points in red section do not impact robot traversability and are discarded; points in green and blue sections are further processed. (b) Horizontal filter: only points within the green angle sector are considered of relevance for the ongoing task.

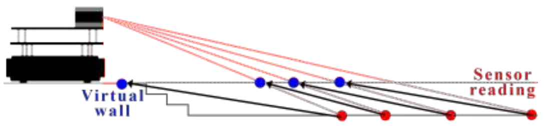


Fig. 4: Stairs recognition: the red point represents the sensor's reading. In this case, the floor in front of the robot can also be seen within the FoV. Therefore, a virtual wall is created projecting sensor readings up to the floor border (the blue point nearest to the robot).

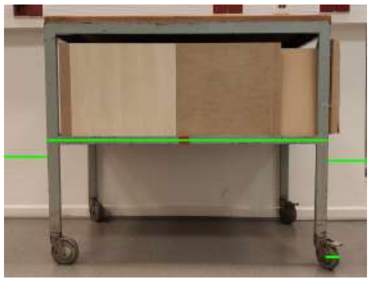


Fig. 5: Selection of nearest points for each horizontal angle: Only the green points are selected.

define a vector of zeros with a size defined as $360°/R$, where R is the desired horizontal resolution in degrees. Also note that R must not be less than the resolution of the sensor, e.g., for an Astra Camera $0.09375°$, for a Hokuyo UTM-30LX $0.25°$ and for a Velodyne VLP-16 between $0.1°$ and $0.4°$. Then, the output points of the previous processing stage can be inserted into their corresponding positions of this new vector. For instance, in the case of a $360°$ view, a point with angle equal to $0°$ would go to index 0 in the vector, and a point with angle equal to $270°$ would go to index $(360°/R) \cdot (270°/360°)$. This data structure resembles a typical laser scan and can be provided together with the resolution R to a 2D mapping algorithm for further processing.

IV. Experimental Results

To evaluate our approach, we implemented the PFF algorithm in C++ and integrated it into ROS Kinetic Kame. We built traversability maps with our service robot based on the TurtleBot 2 platform (Fig. 1), using *GMapping* [1] to process the output of our PFF and build the map. Besides using different sensors on board of the robot, we also tested two different computers with various processing capabilities: BM, a middle-class machine with a 2 cores at 1.9 GHz, i5 processor and 8 GB of RAM; PM, a high-end machine with a 6 cores at 2.2 GHz, i7 processor and 16 GB of RAM, both running Ubuntu 16.04.

We evaluate our approach by measuring processing times and accuracy of the resulting traversability maps. With respect to processing times, we present the average of 100 iterations for each case, under the same mapping parameters. Two types of environments have been tested, with and without the presence of stairs, to assess the impact of our stair recognition module. In all experiments, the PFF resolution parameter was set to $R = 0.4°$ for the resulting vector that is then provided to the GMapping algorithm. h_f was set to the actual height of the sensor mounted on the robot.

A. PFF Processing Time

We focus first on the performance of PFF and therefore analyze its processing time from when the sensor readings are available until the data are converted to be used by mapping algorithms. The results presented in Table II show that the processing time for the BM using a RGB-D camera is approximately 3 and 5.6 times slower for the option without and with stairs recognition, respectively, in comparison with the times obtained for the 3D LiDAR. This is due to the number of points that must be processed. The Astra camera publishes around 9,000,000 points per second while the VLP-16 LiDAR provides only 300,000 points per second.

[1]https://openslam-org.github.io/gmapping.html

TABLE II: PFF processing times (ms) for different sensor configurations and machines.

Sensor (parameters)	$\theta \times \mathbf{h_r}$	BM	PM	Points x Sample
Environment without stairs (Fig. 7)				
Camera	$60° \times 40\ cm$	36.5	10.9	307,200
VLP-16	$360° \times 40\ cm$	11.6	1.2	30,000
VLP-16	$180° \times 40\ cm$	10.6	1.1	30,000
Environment with stairs (Fig. 8)				
Camera	$60° \times 40\ cm$	73.6	17.3	307,200
VLP-16	$360° \times 40\ cm$	13.1	1.5	30,000
VLP-16	$180° \times 40\ cm$	11.8	1.3	30,000

This shows an advantage in the use of 3D LiDAR sensors for indoor service robots, not only for the processing time but also for the larger horizontal field of view, $360°$ for the VLP-16 against the $60°$ of the Astra camera. Moreover, if the horizontal field of view for the 3D LiDAR is restricted from $360°$ to $180°$, a reduction in computation time is also noticeable but only limited if compared to the number of points that the PFF is able to filter.

B. Full Processing Time

After checking the results of the PFF processing time, we analyze the processing time from when the sensor publishes new measurements until GMapping processes the data provided by the PFF. We compare our solution with different sensor configurations with the results of a 3D mapping algorithm (LOAM [13] with a 3D LiDAR) as well as with a 2D mapping algorithm (GMapping with a 2D LiDAR). This experiment was carried out driving the robot manually around the environments shown in Fig. 7 and Fig. 8, with and without stairs, respectively. The results are reported in Table III. Note that these results include processing time for the mapping algorithm, which depends partially on the number of 3D points provided by the sensor. In this experiment the camera was providing fewer points than the 3D LiDAR, and its full mapping process is faster. On the other hand, the 2D liDAR provides more points due to the resolution of the sensor, so it takes more time to process the information.

In the case of LOAM, it can be observed that the processing time for the BM computer is 10 times slower than for PM. On the other hand, the 2D solution for BM is 1.34 times slower than for PM. For the experiments with the PFF, a significant decrease in computation time can be noticed, beating LOAM or the 2D solution in all configurations. Moreover, PFF with the BM is on average only 1.8 times slower than with PM. Thus, PFF improves the efficiency of the mapping procedure, making it possible on middle-class machines, where complex 3D mapping algorithms are impractical. A further increase in speed can be obtained by

TABLE III: Full processing times (ms) on different machines and for different techniques. In LOAM, the parameters indicate the horizontal and vertical FoV of the 3D LiDAR; in Hokuyo, the horizontal FoV of the 2D LiDAR; For the LOAM and the Hokuyo versions, since they do not filter points, results were the same for both scenarios, with and without stairs; in PFF the two parameters are θ and h_r, respectively.

Technique	**Parameters**	**BM**	**PM**	**Sample Time**
VLP-16 + LOAM [13]	$360^\circ \times 30^\circ$	2047	202	100 ms
Hokuyo + GMapping	270°	320	236	25 ms
Environment without stairs (Fig. 7)				
Camera + PFF + GMapping	$60^\circ \times 40\ cm$	56	31	33 ms
VLP-16 + PFF + GMapping	$360^\circ \times 40\ cm$	218	139	100 ms
Environment with stairs (Fig. 8)				
Camera + PFF + GMapping	$60^\circ \times 40\ cm$	75	33	33 ms
VLP-16 + PFF + GMapping	$360^\circ \times 40\ cm$	203	122	100 ms

restricting the FoV (θ parameter) if the application permits.

C. Traversability Results

The main idea of the PFF is to consider the robot height and the presence of stairs in order to determine the traversable areas. To validate this, we placed our robot static in an environment with 3 different obstacles at fixed locations: a chair, a box and a table (Fig. 6). Then, we experimented how the traversability maps would be generated for a fixed value of $\theta = 180^\circ$ and different values of h_r, corresponding to the different robot platforms in our lab.

For the detection of the chair, both PFF with $h_r = 40\ cm$ and $h_r = 25\ cm$ (Fig. 6, bottom-left and bottom-right), with a 3D LiDAR, are able to identify the base of the chair that prevents the robot from passing through. On the contrary a 2D LiDAR ($h_f = 25\ cm$) can only observe the vertical pole of the chair (Fig. 6, top-right). As for the box, the 2D LiDAR is unable to see it because the sensor is located at a height greater than the height of the box (Fig. 6 top-right), while the two options with the PFF can detect it (Fig. 6, bottom). Regarding the table, the PFF option with $h_r = 25\ cm$ only detects the legs of the table (Fig. 6, bottom-right) because the robot would then be able to pass under it. However, when we set $h_r = 40\ cm$ the whole table is shown as an obstacle (Fig. 6, bottom-left), as the robot can not go under the table. On the other hand, a 2D LiDAR only detects the legs of the table, which are at the observable height of the sensor.

D. Mapping Results

We finally tested the full processing chain including the building of the traversability map. For this, we navigated our robot manually throughout a static environment to create 2.5D maps with $\theta = 180^\circ$ [2], h_f = 25 cm and different values of h_r, using PFF with a 3D LiDAR. We then compared the results with those obtained with a 2D LiDAR. This experiment was carried out in a room (Fig. 7) with tables

[2]We did not use the full FoV of the LiDAR due to occlusions by other physical components on the robot.

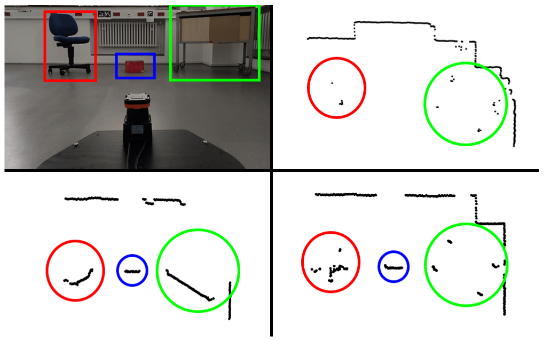


Fig. 6: Results of the laser scans for different configurations. Top-left, static scenario with a chair (marked red), a box (marked blue) and a table (marked green). Top-right, 2D laser. Bottom-left, PFF with a 3D LiDAR and $h_r = 40\ cm$. Bottom-right, PFF with a 3D LiDAR and $h_r = 25\ cm$.

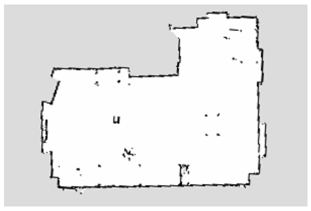
TABLE IV: Traversable area for an office-like environment.

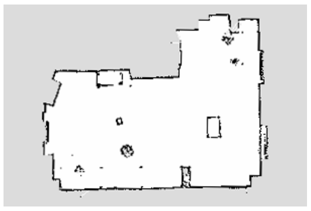
Sensor	**h_r (cm)**	**Traversable Square Meters**	**Difference w.r.t. 2D Map**
Environment without stairs (Fig. 7)			
VLP-16 (Fig. 7 (a))	25	64.7	3.14%
VLP-16 (Fig. 7 (b))	40	62.1	7.03%
VLP-16 (Fig. 7 (c))	100	53.5	19.91%
2D LiDAR (Fig. 7 (d))	–	66.8	–
Environment with stairs (Fig. 8)			
VLP-16 (Fig. 8 (b))	25	6.46	64.93%
2D LiDAR (Fig. 8 (c))	–	18.42	–

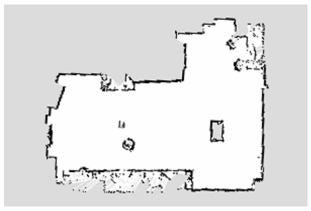
of different heights, chairs, boxes and cabinets. Regardless of the h_r value, the traversable area computed through the PFF is always lower compared with the result obtained by a 2D sensor, as it is seen in Table IV. This is due to the obstacles that the 2D sensor is not able to correctly perceive. In this sense, the difference with respect to the mapping based on a 2D-only sensor reports the obtained increase in mapping accuracy by reporting the percentage of the area that would be incorrectly recognized as traversable without a 2.5 mapping. Table IV also shows how this difference changes for different robot heights in comparison to a pure 2D solution that would not be able to consider this information. This can be seen more clearly by comparing Fig. 7(a) with Fig. 7(d), where in the bottom left there is a wide section of tables that the 2D LiDAR cannot detect. If the robot had to move in that area, it would collide against those objects with a standard 2D map. Also, in an environment with stairs (Fig. 8 (a)), where the 2D LiDAR is unable to see the stairs (Fig. 8 (c)), the PFF correctly creates a virtual obstacle, preventing the robot from falling down (Fig. 8 (b)). For our case study, the difference in the accuracy of the correct traversability map reaches nearly a third of the whole area.

V. Conclusions

We presented the Point-cloud Fast Filter (PFF), an efficient solution that, combined with a standard SLAM algorithm, can build traversable maps for service robots, specifically targeting platforms without high computational capabilities. We focused on indoor, office-like environments, which are


(a)


(b)


(c)

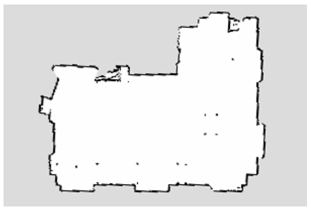
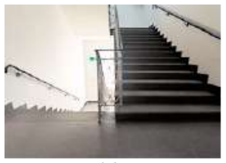
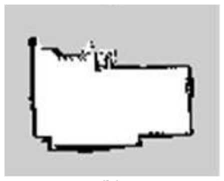

(d)

Fig. 7: Traversability map for an office-like scenario: (a) PFF, VLP-16 and robot height = 25 cm, (b) PFF, VLP-16 and robot height = 40 cm, (c) PFF, VLP-16 and robot height = 100 cm and (d) Hokuyo 2D LiDAR.


(a)


(b)

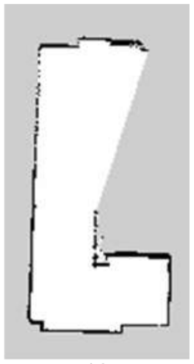

(c)

Fig. 8: Traversability map for a scenario with stairs: (a) Scenario, (b) PFF with a 3D LiDAR, (c) 2D LiDAR.

at the same time structured but with obstacles at different heights (e.g., chairs or tables) that may prevent robots from passing through. Our algorithm filters out the input data from 3D sensors taking into account robot dimensions and field of view of interest, in order to make the processing more efficient. It also includes a module to detect stairs/holes (no ramps) or similar obstacles that are typical in our considered scenarios and not detected by 2D LiDAR sensors. Moreover, our PFF is sensor and mapping agnostic in the sense that it can be easily integrated with different sensors such as RGB-D cameras or LiDARs, as well as with standard SLAM algorithms.

Our results show an improvement in processing times compared to other 2D and 3D mapping techniques. This allows the service robot to reduce its hardware requirements for the mapping task, and devote more resources to additional services. Moreover, this reduction in the processing time does not come at cost of safety, since the traversability maps are enough to avoid collisions for the robot and environment at hand. As future work, we plan to test our method in more dynamic environments with people moving around and test the long-term navigability of the robot.

References

[1] N. Perez-Higueras, R. Ramon-Vigo, I. Perez-Hurtado, J. Capitan, F. Caballero, and L. Merino, "A social navigation system in telepresence robots for elderly," in *Using social robots to improve the quality of life in the elderly, Workshop on the 8th International Conference on Social Robotics*, Kansas City, USA, 2016.

[2] J. Biswas and M. M. Veloso, "Localization and navigation of the cobots over long-term deployments," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1679–1694, 2013.

[3] J. Messias, R. Ventura, P. Lima, J. Sequeira, P. Alvito, C. Marques, and P. Carrio, "A robotic platform for edutainment activities in a pediatric hospital," in *2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, May 2014, pp. 193–198.

[4] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: a survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 55–81, 2015.

[5] J. M. Santos, D. Portugal, and R. P. Rocha, "An evaluation of 2D SLAM techniques available in robot operating system," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2013, pp. 1–6.

[6] S. Stiene and J. Hertzberg, "Virtual range scan for avoiding 3D obstacles using 2D tools," in *2009 International Conference on Advanced Robotics*, June 2009, pp. 1–6.

[7] D. Holz, D. Droeschel, S. Behnke, S. May, and H. Surmann, "Fast 3D perception for collision avoidance and SLAM in domestic environments," in *Mobile robots navigation*. IntechOpen, 2010.

[8] F. Nex and F. Remondino, "UAV for 3D mapping applications: a review," *Applied Geomatics*, vol. 6, no. 1, pp. 1–15, 2014.

[9] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D mapping with an RGB-D camera," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, Feb 2014.

[10] C. Li, H. Wei, and T. Lan, "Research and implementation of 3D SLAM algorithm based on kinect depth sensor," in *International Congress on Image and Signal Processing, BioMedical Engineering and Informatics*, Oct 2016, pp. 1070–1074.

[11] J. Biswas and M. Veloso, "Depth camera based indoor mobile robot localization and navigation," in *IEEE International Conference on Robotics and Automation*, May 2012, pp. 1697–1702.

[12] O. Wulf, K. O. Arras, H. I. Christensen, and B. Wagner, "2D mapping of cluttered indoor environments by means of 3D perception," in *IEEE International Conference on Robotics and Automation*, vol. 4, 2004, pp. 4204–4209.

[13] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, 2014, p. 9.

[14] K. Yoneda, H. Tehrani, T. Ogawa, N. Hukuyama, and S. Mita, "Lidar scan feature for localization with highly precise 3-D map," in *IEEE Intelligent Vehicles Symposium Proceedings*, 2014, pp. 1345–1350.

[15] R. W. Wolcott and R. M. Eustice, "Fast lidar localization using multiresolution gaussian mixture maps," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2814–2821.

[16] A. A. S. Souza and L. M. G. Gonalves, "2.5-dimensional grid mapping from stereo vision for robotic navigation," in *Brazilian Robotics Symposium and Latin American Robotics Symposium*, 2012, pp. 39–44.

[17] D. De Gregorio and L. Di Stefano, "Skimap: An efficient mapping framework for robot navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2569–2576.

[18] F. Yuan, A. Swadzba, R. Philippsen, O. Engin, M. Hanheide, and S. Wachsmuth, "Laser-based navigation enhanced with 3D time-of-flight data," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 2844–2850.

[19] SLAMTEC. RP LiDAR A3 specifications. [Online]. Available: https://www.slamtec.com/en/Lidar/A3Spec

[20] Orbbec. Astra camera specifications. [Online]. Available: https://orbbec3d.com/product-astra-pro/

[21] Velodyne. Velodyne VLP-16 specifications. [Online]. Available: https://velodynelidar.com/vlp-16.html